Draft	E. Prodromou
	B. Vibber
	J. Walker
	Z. Copley
	StatusNet Inc
	August 30, 2010

TOC

OStatus 1.0 Draft 2

Abstract

OStatus lets people on different social networks follow each other. It applies a group of related protocols (PubSubHubbub, ActivityStreams, Salmon, Portable Contacts, and Webfinger) to this problem in what we believe is a simple and obvious way.

OStatus is a minimal specification for distributed status updates or microblogging. Many social applications can be modelled with status updates, however. Practically any software that generates RSS or Atom feeds could be OStatus-enabled. Travel networks, event invitation systems, wikis, photo-sharing systems, social news sites, social music sites, podcasting servers, blogs, version control systems, and general purpose social networks would all be candidates for OStatus use.

The authors recognize that most of this protocol suite was intended to work together in a natural way. They claim no originality or creativity in connecting their use. The authors hope that the obviousness of some parts of this specification are an argument in favor of its use.

Table of Contents

TOC

1. License

TOC

2. Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in **[RFC2119]**.

XML namespace prefixes used in this document:

atom:	http://www.w3.org/2005/Atom
activity:	http://activitystrea.ms/spec/1.0/
schema:	http://activitystrea.ms/spec/1.0/
ostatus:	http://ostatus.org/schema/1.0/
thr:	http://purl.org/syndication/thread/1.0

TOC

3. Definitions

User:
: A person or organization, or a program that mimics a person or organization ("bot").

Group:
: A set of users.

Update:
: Notification of a change in a user's status or an action taken by a user. Updates can be automatically generated by software or composed by the user.

Feed:
: A list of related updates. For example, a feed may consist of all updates by a user, all updates by users that are members of a group, updates that match a search query.

Publisher:
: A user or group that makes their updates available for others.

Subscriber:
: A user who receives updates from one or more publishers. Also known as "follower".

Follow:
: Receive updates from a publisher.

publisher server:
: the Web-accessible server that the publisher uses to make updates available.

subscriber server:
: the Web-accessible server that the subscriber uses to receive updates.

TOC

4. Requirements

These are the parameters of the problem we wish to solve.

- An update may be represented with plain text in UTF-8 encoding.
- An update may be represented with HTML.
- An update may include one or more attached files.
- An update may be a response to another update.
- An update can be a forwarded or shared copy of another update. ("repeat", "retweet").
- An update can be about a topic.
- An update can be directed to the attention of a particular recipient. ("mention").

- An update can be related to a location on Earth.
- Anyone can mark an update as a "favorite", or remove that mark.
- Users have a unique identity.
- Users have profile information like name, location, nickname or username, bio, related URLs.
- Users can be represented with a image ("avatar").
- Subscriber can receive Publisher's updates very soon after publication. ("Real time", "near real time") (Upshot: we want push.)
- Publisher server can store a list of all Subscribers for a given Publisher, including identities and profile data.
- Subscriber server can store a list of all Publishers for a given Subscriber, including identities and profile data.
- Publisher server can store a list of all responses to an update by a publisher.
- Publisher server can store a list of all forwarded copies of an update by a publisher.
- Publisher server can store a list of all "favorites" of an update by a Publisher.
- Servers can determine update context and metadata without parsing the plain text according to a particular syntax.
- Subscribers can subscribe to a feed with multiple authors, such as a group, a search feed, or a tag feed.

---

5. Out of scope

The following common elements of microblogging and status update systems are out of scope for OStatus.

Microblogging syntax:
: Microblogging servers use a number of different idioms for adding metadata to an update. Notable examples are hashtags (#hashtag) and @-replies (@username). We defer to microsyntax.org for developing these idioms and encourage innovation. Publisher and subscriber servers should be able to process received updates without parsing the plain text content.

Client API
: Microblogging systems typically allow many forms of communication between the publisher and the publisher server as well as the subscriber and the subscriber server. Some examples: Web interface, email, IM, SMS, and a Web API. By analogy with email, OStatus is to client interfaces as SMTP is to IMAP or POP.

Private messaging:
: OStatus does not directly address sending 1-to-1 private messages or private streams.

Social graph:
: OStatus does not specify a static representation of the social graph nor a protocol for retrieving that representation for a user. We defer to FOAF or XFN where this is required.

---

6. Update representation

Updates are represented as **Activities** [activitiesinatom] in **Atom** [RFC4287]. Typical updates would be represented in the **default Activity Schema** [activityschema] with activity verb "Post" and activity object type "Note", "Status" or "Comment". Servers SHOULD support these default types, and MAY support others.

A spatial location for the update object should be encoded as **GeoRSS** [georss] element as part of the activity.

Attachments to an update should be represented as enclosures.

If an update represents a notice made in reply to another, this should be represented using the in-reply-to element from **[RFC4685]**.

Tag information should be represented as Atom categories.

OStatus defines two custom extensions to Activity streams.

link[@rel=ostatus:attention]/@href
: When an update is directed to the attention of someone, or mentions someone, this link stores that user's URI.

link[@rel=ostatus:conversation]/@href
: When an update is part of a distributed conversation, this is the URI of that conversation.

7. User representation

Users are identified by URIs.

Users are represented as activity objects of the appropriate object type. A typical user would have type "Person".

In the absence of additional information, a default Atom or RSS feed without additional Activity data will be represented as with activity verb "Post" and activity object type "Note", with the Atom author as the implied activity actor. In this case, the author element MUST have a unique identifier, that is, either the <email> or <uri< elements must be present.

Subscribers MUST search for the URI in the following order: the Activity object's <id> element if available, else the Atom author's <uri> element if available, else the Atom author's <email> element treated as a Webfinger acct: URI. Publishers MAY use a profile URL as the URI, but if so MUST ensure that they are stable permalinks.

Users SHOULD have a profile URL, which SHOULD be an HTTP or HTTPS reference to an HTML page including discovery information for the user's feed. The profile URL SHOULD be represented as a link[@rel=alternate,@type=text/html] in the Activity subject, actor, or object item, otherwise the URI MAY be used if it is an HTTP or HTTPS URL.

To provide additional profile information, the author or actor element can be extended with optional **Portable Contacts** [poco]-structured data. Some elements worth highlighting:

preferredUsername:
: A login or username.

displayName:
: Full name. If present, it SHOULD be identical to the author/name or actor/title value.

note:
: A free text description of the user.

address/formatted:
: A free text description of the user's current location or usual location.

urls[type=home]/value:
: URL of a representative page for the user. Note that this is distinct from the link[@rel=alternate,@type=text/html] required for activity actors.

Subscribers MAY use other sources of profile information such as microformats on the profile page, but are not required to.

8. Feed representation

Feeds are Activity feeds per **[activitiesinatom]**.

All feeds SHOULD have an activity:subject element. User feeds SHOULD have a single activity:subject or atom:author. Group feeds SHOULD have a single activity:subject representing the group.

All feeds MUST have a link[@rel=hub] to identify a PubSubHubbub endpoint URL to establish update subscriptions.

All feeds SHOULD have a link[@rel=salmon] to identify the Salmon endpoint URL to receive replies and notifications.

9. Update publication

The publisher server uses **PubSubHubbub** [push] to notify subscribers of new updates.

If multiple subscribers using the same subscriber server subscribe to the same publisher, the subscriber server SHOULD use the one PubSubHubbub subscription for all of them.

10. User notification

Servers use **Salmon** [salmon] to post social events to users or groups. Servers MUST NOT assume that updates that were modeled as activities in a PubSubHubbub-enabled feed were received by the user.

Note that sender and receiver of a notification MAY NOT have a subscriber-publisher relationship.

Important social events and related activities:

Attention:
: When a user posts an update to the attention of another user or a group, their server sends the update to the salmon endpoint of that user or group. The update MUST have the ostatus:attention link set to the URI of the receiving user or group.

Reply:
: When a user posts an update in reply to another update, their server SHOULD post it as a Salmon entry to the author of the original update. The reply MUST have the thr:in-reply-to element set to the URI of the original notice. The reply MAY have the ostatus:attention link set to the URI of the original update's author.

Subscribe:
: When a subscriber starts following a user, the subscription server sends a schema:follow activity.

Unsubscribe:
: When a subscriber stops following a user the subscription server sends an ostatus:unfollow activity.

Join:
: When a subscriber starts following a group, the subscriber server sends a schema:follow activity.

Leave:
: When a subscriber stops following a user the subscriber server sends an ostatus:leave activity.

Favorite:
: When a user marks another user's update as a favorite, their server sends an update with verb schema:favorite.

Unfavorite:
: When a user has marked another user's update as a favorite, and removes that mark, their server sends an update with verb ostatus:unfavorite.

Repeat:
: When a user publishes a copy of an update to their own feed, their server sends an update with verb schema:share to the original verb.

11. Discovery

Information necessary to establish a subscription or post a notification can be determined from the feed for a user or group. The profile data, salmon endpoint, and PubSubHubbub hub are all encoded in the feed itself.

There are three ways to discover the location of the feed.

1. Direct input from the user.
2. From Link/Rel=http://schemas.google.com/g/2010#updates-from element of a related XRD file for the user.
3. From a link[@rel=alternate,@type=application/atom+xml] element in the HTML of a profile page.

There are two ways to get a profile page URL.

1. Direct input from a user.
2. From Link/Rel=http://webfinger.net/rel/profile-page of a related XRD file for the user.

There are two ways to get a related XRD file for a user.

1. Given an acct: URI like nickname@example.com, use **[Webfinger]** to discover the user's XRD file.
2. From a Link: HTTP header for the related profile page, use **[LRDD]**.

The user's XRD document MAY have an additional link template with Rel equal to http://ostatus.org /schema/1.0/subscribe to indicate the endpoint to use for initiating a subscription on the user's subscription server. The template should take a single argument, uri, for the account to subscribe to.

12. Usage scenarios

These are some rough scripts for some important processes in this system. They are non-normative.

12.1. Subscription from Subscriber server

1. Subscriber notifies Subscriber server of a feed to which she wants to subscribe (e.g., by providing a HTTP URL, a Webfinger account, or choosing from a list of known feeds).
2. The subscriber server discovers the URL of the feed.
3. If the Subscriber server already has a subscription to this feed, it skips to step 6.
4. Subscriber server downloads the feed using HTTP.
5. Subscriber server checks for a PubSubHubbub URL in the feed document. If there's no PubSubHubbub URL, the server MAY end the subscription. Any other subscription, e.g. one using polling, is out-of-band for OStatus.
6. Subscriber server subscribes to the feed using PubSubHubbub subscription mechanism.
7. Subscriber server checks the feed for a Salmon endpoint. If none exists, the process is complete.
8. Subscriber server uses Salmon to push a representation of the subscription to the Publisher server.

12.2. Subscription from Publisher server

1. Subscriber provides a WebFinger account to the Publisher server (say, in an HTML form).
2. Publisher server uses WebFinger to discover the subscription URITemplate for this user.
3. Publisher server substitutes Publisher's feed URL into URITemplate.
4. Publisher server redirects user's browser to substituted URL.
5. Subscription continues from step 1 of previous case.

13. References

[LRDD]	Hammer-Lahav, E., “**Link-based Resource Descriptor Discovery**,” March 2010.
[RFC2119]	**Bradner, S.**, “**Key words for use in RFCs to Indicate Requirement Levels**,” BCP 14, RFC 2119, March 1997 (**TXT**, **HTML**, **XML**).
[RFC4287]	**Nottingham, M., Ed.** and **R. Sayre, Ed.**, “**The Atom Syndication Format**,” RFC 4287, December 2005 (**TXT**, **HTML**, **XML**).
[RFC4685]	Snell, J., “**Atom Threading Extensions**,” RFC 4685, September 2006 (**TXT**).
[Webfinger]	“**the WebFinger protocol**.”
[activitiesinatom]	Atkins, M., Recordon, D., Messina, C., Keller, M., Steinberg, A., and R. Dolin, “**Atom Activity Extensions (Draft)**,” March 2010.
[activityschema]	Atkins, M., Recordon, D., Messina, C., Keller, M., Steinberg, A., and R. Dolin, “**Atom Activity Base Schema (Draft)**,” March 2010.
[georss]	“**GeoRSS-Simple**.”
[poco]	**Smarr, J.**, “**Portable Contacts 1.0 Draft C**,” August 2008.
[push]	**Fitzpatrick, B.**, **Slatkin, B.**, and **M. Atkins**, “**PubSubHubbub Core 0.3 -- Working Draft**,” February 2010.
[salmon]	**Panzer, J.**, “**The Salmon Protocol**,” February 2010.

Authors' Addresses

Evan Prodromou
StatusNet Inc
Email: evan@status.net
URI: http://evan.status.net/

Brion Vibber
StatusNet Inc
Email: brion@status.net

James Walker

StatusNet Inc
Email: james@status.net

Zach Copley
StatusNet Inc
Email: zach@status.net